À la découverte des microcontrôleurs PIC

(Septième partie)

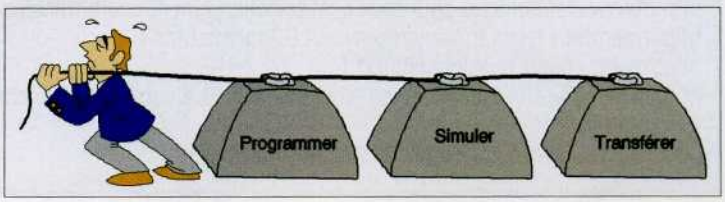
Dans ce numéro, nous allons détailler le transfert d'un fichier compilé vers la mémoire d'un PIC en nous intéressant plus particulièrement à l'aspect matériel et aux programmateurs. Nous reviendrons sur la simulation avec le logiciel MPLAB.

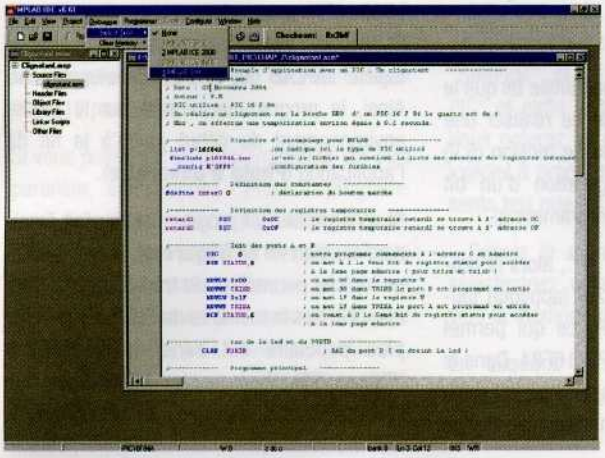
Dans notre dernier numéro, nous avons détaillé puis compilé un programme. Nous allons aborder aujourd'hui un aspect intéressant, celui de la simulation. L'intérêt de simuler un programme est bien sûr très pédagogique car comme nous le verrons, nous allons pouvoir faire du "pas à pas " et visualiser le fonctionnement de tous les registres constituant le PIC. De plus, nous allons pouvoir vérifier le fonctionnement du programme avant de le transférer vers la mémoire du microcontrôleur. Cela nous permettra d'économiser du temps et également d'augmenter la durée de vie de notre PIC puisque la mémoire flash de celui-ci n'est pas, comme vous le savez, éternelle...

Nous allons simuler le clignotant, programme que nous avons détaillé dans la dernière leçon.

Pour passer un programme en mode simulation, vous devez, depuis le logiciel MPLAB, ouvrir votre projet tel que nous l'avons vu dans notre dernière leçon, puis, depuis le menu " "Debugger", sélectionner "Select Tools" et "MPLAB SIM" (**Figure 2**).
Une fois le mode simulation déclaré, il faut placer les stimuli qui correspondent aux entrées du PIC. Pour ce faire, allez dans le menu "Debugger" et sélectionnez "Stimulus Controller". Vous devez alors avoir une fenêtre supplémentaire correspondant à la **Figure 3**. C'est dans cette fenêtre que nous allons déclarer les entrées qui seront alors modifiables par un clic de souris.
Dans notre programme "clignotant", nous n'avons en fait qu'une seule entrée à piloter, c'est le bouton marche-arrêt dont l'état permet au programme soit de se lancer, soit de se reboucler sur le test de cet interrupteur. C'est la broche RA0 qui a pour rôle de recevoir cet interrupteur, nous allons donc déclarer celle-ci dans la fenêtre de stimuli.
Pour déclarer la broche RA0 en entrée de stimuli, cliquez sur le bouton "ADD ROW", puis, dans le type, sélectionnez

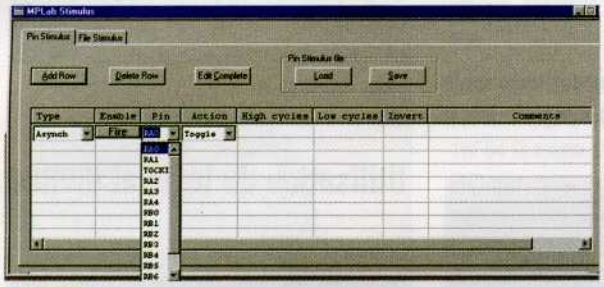
1 ***Phases de développement***



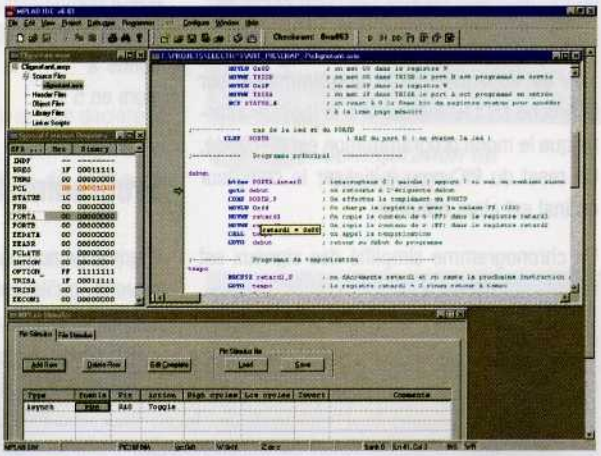


2 ***Sélection du mode simulation***

3 ***Fenêtre de déclaration des stimuli***



4 ***Sélection des entrées de stimuli***



5 ***Phase de simulation***

"Asynch" ainsi que la broche concernée "RA0" ainsi que dans la dernière liste déroulante, l'option "Toggle". Le bouton "Fire" est là pour valider désormais vos clic de souris. Sachant que nous avons pris l'option "Toggle", chaque appui sur "Fire" fera changer l'état de la broche RA0. Bien sûr, si nous avions plusieurs entrées à simuler, il suffirait de refaire un " Add Row" et de recommencer les étapes détaillées ci-dessus (**Figure 4**).

Nous venons de mettre en place notre entrée de validation RA0 et nous pouvons dès maintenant commencer la simulation. Pour ce faire et afin de visualiser l'action sur le bouton "Fire" qui simulera notre entrée RA0, nous allons ajouter une vue d'écran. Cliquez dans le menu "View" puis sélectionnez "Spécial Function Registers" afin de visualiser l'état du port A (pour visualiser l'entrée RA0 entre autre). Cet écran est représenté **figure 5**.

Sélectionnez maintenant le menu "Debugger" puis cliquez sur "Animate" dans la fenêtre où se trouve le source du programme. Il apparaît une flèche verte qui se positionne au rythme de l'horloge sur l'instruction en cours d'exécution (**Figure 5**). Si vous n'avez pas appuyé encore sur le bouton "Fire", la flèche oscille entres les deux instructions :

btfss PORTA,inter0 ; interrupteur 0 (marche) appuyé ? si oui on continu sinon on retourne à l'étiquette debut
goto debut ;

Appuyez maintenant sur le bouton "Fire" et notez au passage le changement d'état du registre PORTA de la fenêtre Spécial Function Registers qui passe de 0000 0000 à 0000 0001 puisque la broche RA0 vient de passer à 1 par le clic de souris. À ce moment, le programme continue de se dérouler. L'appui sur le bouton marche-arrêt vient d'être simulé : pratique non ?

Pour arrêter le mode animation, appuyez sur la touche F5 ou bien allez dans le menu "Debugger" puis cliquez sur "Halt".

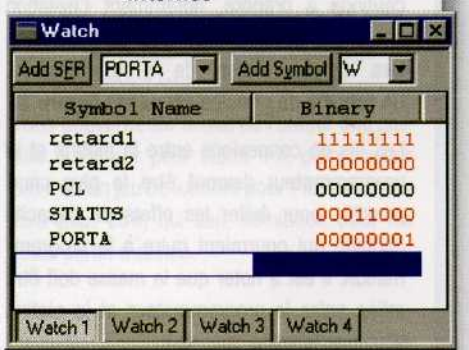
Pour faire un reset de l'animation, appuyez sur la touche F6 ou bien allez dans le menu "Debugger" puis cliquez sur "Reset" et sur "Processor Reset".

Depuis le menu view, vous pouvez ajouter des écrans composés de différents registres internes du PIC, y compris ceux que vous avez déclarés dans votre programme. Par exemple, depuis le menu "View", cliquez sur "Watch" puis dans la liste déroulante "Add Symbol" sélectionnez le registre "retard1" et validez sur le bouton "Add Symbol". Comme vous l'avez constaté, vous pouvez visualiser tous les registres internes au PIC ce qui est très pédagogique pour en comprendre le fonctionnement (**figure 6**).

Nous voici arrivés au stade où nous avons écrit notre programme, nous l'avons simulé et tout se passe bien : la simulation correspond à nos attentes…

Nous pouvons maintenant transférer le fichier compilé (clignotant.hex) vers la mémoire du PIC.

6 ***Visualisation de différents registres internes***



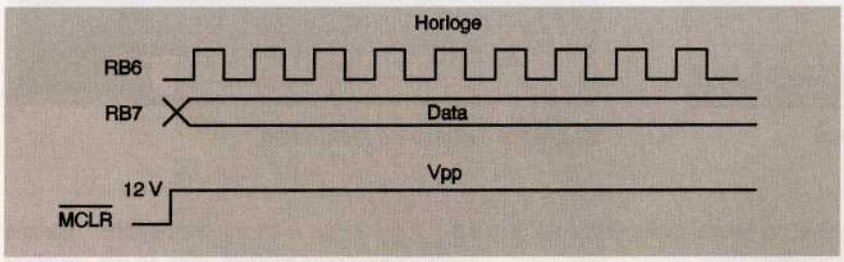
Phase de programmation

Trois signaux sont nécessaires pour programmer un PIC, en effet la programmation nécessite un signal d'horloge. Celui-ci sera appliqué sur la broche RB6 du PIC, quant aux données, elles transiteront sur la broche RB7.

Il est à noter que cette broche sera bidirectionnelle puisque l'on pourra également lire le programme contenu dans la mémoire flash du PIC ainsi que dans la mémoire EEPROM.

Lors de la programmation, une tension de 12 V est appliquée par le programmateur sur la broche MCLR/ indiquant à la logique interne que le mode programmation est demandé. Un reset du PIC pour initialiser le compteur ordinal est alors effectué.

Un chronogramme simplifié des signaux est représenté **figure 7**.



7 *Les signaux de programmation*

Programmation ICP

La programmation du PIC nécessite soit d'enlever le PIC du montage sur lequel il est implanté et de l'insérer dans le programmateur, soit de réaliser la programmation in-situ (directement sur la platine d'application). On parlera alors de programmation ICP (In-Circuit Programming). Le mode ICP est très pratique puisque l'on ne doit pas enlever le PIC du montage pour le programmer ce qui est appréciable. Lorsque l'on fait de la mise au point, il y a quand même quelques précautions à prendre, notamment l'isolation des broches RB6 et RB7 qui sont utilisées lors de la phase de programmation. Un schéma de principe est donné **Figure 8**.

Les fils de connexions entre la platine et le programmateur devront être le plus court possible pour éviter les effets de capacité parasite qui pourraient nuire à la programmation. Il est à noter que la masse doit être reliée entre le programmateur et la platine ainsi que le +5 V.

Mode LVP

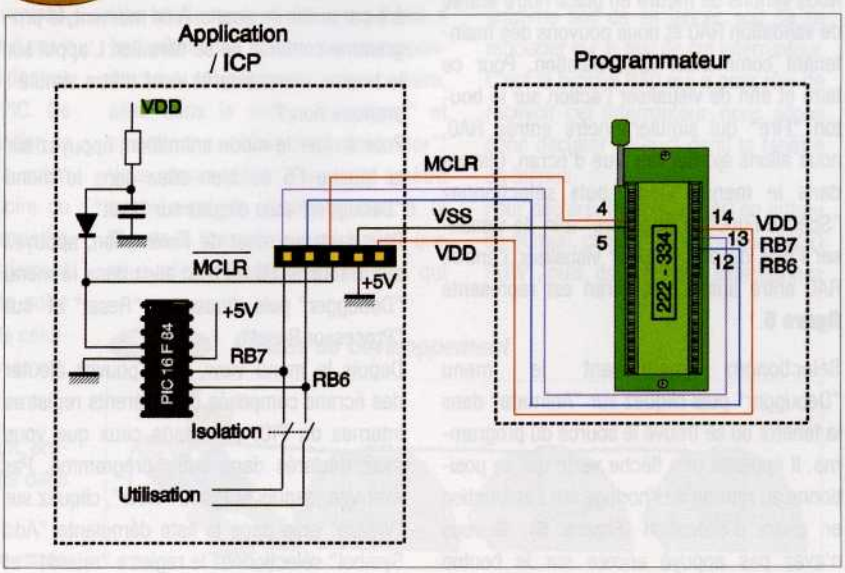
Sur d'autres PIC de la même famille tel que le PIC 16F628, il est possible de réaliser une programmation en mode basse tension (5 V) et ceci suivant la configuration d'un bit nommé LVP (low voltage programming).

Si ce bit est positionné à " 0 ", alors la programmation se fait en +12 V (appliqué toujours sur la broche MCLR) ce qui permet d'être compatible avec le PIC 16F84. Dans le cas du PIC 16F628 ou bien si ce bit est positionné à " 1 ", la programmation se réalise alors en 5 V.

Programmateurs

Il existe de nombreux programmateurs sur le marché, qu'ils soient sous forme de kits à monter ou bien déjà câblés (voir nos annonceurs dans les pages de publicité). Nous en avons également proposés dans de précédents articles.

8 *Programmation in-situ*



MICROCHIP propose entre autre le PICSTART PLUS dont l'intérêt est d'être reconnu par le logiciel MPLAB qui sert de développement. Ainsi, le programmeur reste sur le même environnement du début jusqu'à la fin de l'application (**Photo d'ouverture**).

Si vous utilisez un programmateur fait "maison" ou bien en kit (**Figure 9**), il vous faudra un logiciel permettant de transférer le fichier compilé vers la mémoire du PIC. Le logiciel le plus populaire est sans doute ICPROG (**Figure 10**). Vous pouvez télécharger une version de ce logiciel sur le site *WWW.ICPROG.COM*.

Lors de l'achat du programmateur, préférez les programmateurs pilotés sur port série ou USB, sachant que le port parallèle est de moins en moins utilisé (surtout avec les versions récentes de WINDOWS).

Utilisation du logiciel ICPROG

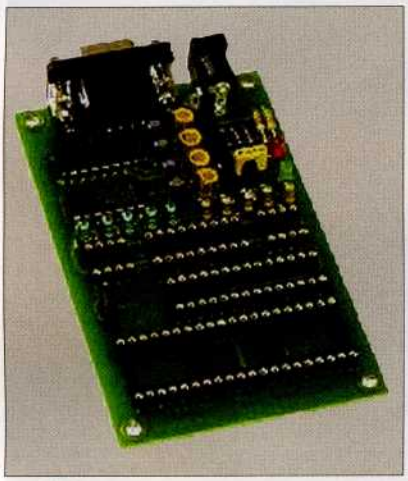
Le logiciel ICPROG possède une interface très conviviale. En quelques clic de souris, nous arrivons rapidement à transférer le fichier compilé vers le PIC.

Pour procéder à une programmation, il faut dans un premier temps :

- Définir la partie "Hardware" : si vous possédez un programmateur de type "série", allez dans le menu "Settings" puis "Hardware" et sélectionnez "JDM PROGRAMMER" dans la liste déroulante puis "DIRECT I/O" si vous travaillez sous WIN 9x

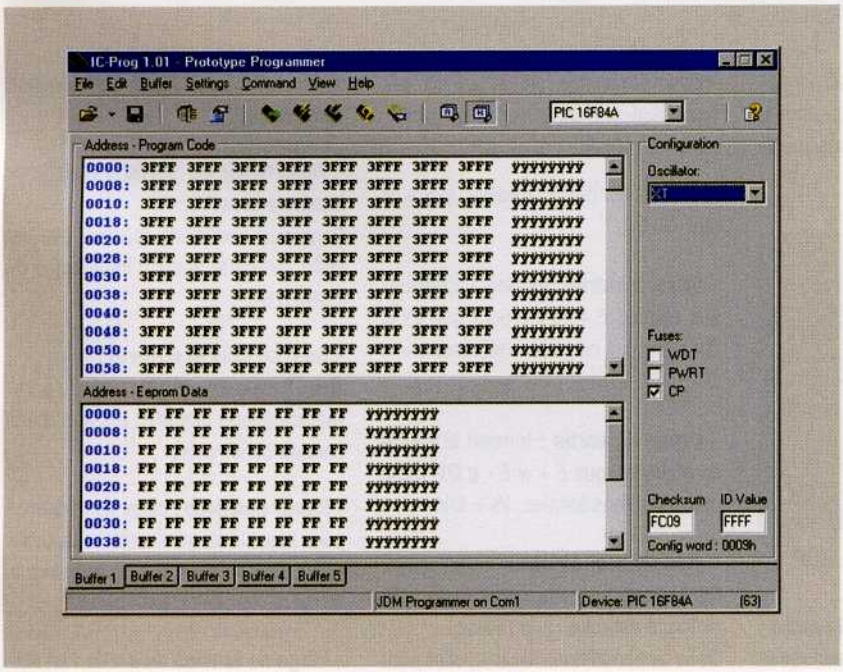
(**Figure 11**) ou bien "Windows API" pour les versions de Windows XP. Enfin, il faut bien sûr sélectionner le port de communication "COM1" par exemple.

Si vous possédez un programmateur sur port parallèle, sélectionnez "Propic2 programmer" puis "Lpt1" pour le port. Attention, certains signaux sont peut être inversés selon le type de programmateur. Dans ce cas, il faudra cocher les cases correspondantes pour les cinq signaux (MCLR,DATA in, DATA out, Clock, VCC). Le mieux consiste bien sûr à s'appuyer sur la documentation du programmateur, en général la configuration de base est indiquée.



9 ***Programmateur de PIC***

10 ***Vue du logiciel ICPROG***



- Sélectionner le type de composant à programmer : Pour ce faire, allez dans le menu "Settings" puis "Device" puis "Microchip PIC" et enfin le type de PIC à programmer. Vous noterez au passage les possibilités du logiciel à programmer de nombreux composants tels que les mémoires I2C.

- Depuis le menu "File" puis "Open file", sélectionnez votre fichier .hex à transférer vers la mémoire du PIC.

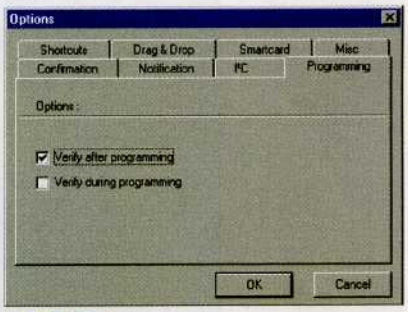
En cliquant dans le menu "View" puis "Assembler", vous pouvez visualiser le code en assembleur.

- Vous pouvez, si vous n'avez pas déclaré la variable _CONFIG dans votre programme source, redéfinir les "fusibles de configuration".

Leur rôle, nous l'avons déjà vu dans une précédente leçon, consiste à choisir pour le PIC 16F84 (il existe d'autre fusibles pour certains PIC, vous pouvez les visualiser en sélectionnant un autre PIC dans la liste) un mode pour l'horloge, la protection pour la relecture, la temporisation au démarrage ainsi que l'utilisation du watchdog. La configuration consiste à cocher ou non les cases situées sur la partie gauche de l'écran (frame "configuration") et de sélectionner un type d'horloge dans la liste déroulante.

- Cliquez ensuite dans le menu "Command" puis "Programm all" pour lancer le transfert du fichier .hex vers la mémoire du PIC. Pour la vérification, vous pouvez, soit lancer en manuel une fois la programmation effectuée

11 ***Configuration du "HARWARE"***



12 ***Modes de vérification***

ce mode en cliquant dans le menu "Command" puis "Verify" ou bien sélectionner une vérification automatique en fin ou en cours de programmation depuis le menu "Settings" puis "Options" et l'onglet "Programming" (**Figure 12**).

Pour conclure ce chapitre

Avec cette septième partie, nous avons enfin téléchargé notre programme vers la mémoire du PIC, sans oublier la partie simulation qui peut être bien sûr approfondie afin de bénéficier des possibilités d'apprentissage du fonctionnement interne des registres .

Vous trouverez sur le site de l'auteur une platine à réaliser pour tester vos programmes ainsi qu'un jeu de lumière pour PIC 16F84 qui fera peut être, qui sait, sensation pour les fêtes de fin d'année.

P.MAYEUX
http://perso.libertysurf.fr/p.may